Class 6: R Function Homework

Lana (PID: A17013518)

Section 1: Improving analysis code by writing functions

> A. Improve this regular R code by abstracting the main activities in your own new function. Note, we will go through this example together in the formal lecture. The main steps should entail running through the code to see if it works, simplifying to a core working code snippet, reducing any calculation duplication, and finally transferring your new streamlined code into a more useful function for you.

```r
# (A. Can you improve this analysis code?
df <- data.frame(a=1:10, b=seq(200,400,length=10),c=11:20,d=NA)
df$a <- (df$a - min(df$a)) / (max(df$a) - min(df$a))
df$b <- (df$b - min(df$a)) / (max(df$b) - min(df$b))
df$c <- (df$c - min(df$c)) / (max(df$c) - min(df$c))
df$d <- (df$d - min(df$d)) / (max(df$a) - min(df$d))

df
```

```
           a        b         c  d
1  0.0000000 1.000000 0.0000000 NA
2  0.1111111 1.111111 0.1111111 NA
3  0.2222222 1.222222 0.2222222 NA
4  0.3333333 1.333333 0.3333333 NA
5  0.4444444 1.444444 0.4444444 NA
6  0.5555556 1.555556 0.5555556 NA
7  0.6666667 1.666667 0.6666667 NA
8  0.7777778 1.777778 0.7777778 NA
9  0.8888889 1.888889 0.8888889 NA
10 1.0000000 2.000000 1.0000000 NA
```

```r
# Define a function using range
col <- function(col) {
  (col - min(col, na.rm = TRUE)) / (max(col, na.rm = TRUE) - min(col, na.rm = TRUE))
}

# Apply the function to each column of the dataframe
simpledf <- function(df) {
  for (col in names(df)) {
    df[[col]] <- col(df[[col]])
  }
  return(df)
}

# Create dataframe
df <- data.frame(a=1:10, b=seq(200,400,length=10), c=11:20, d=NA)

df <- simpledf(df)
```

```
Warning in min(col, na.rm = TRUE): no non-missing arguments to min; returning
Inf

Warning in max(col, na.rm = TRUE): no non-missing arguments to max; returning
-Inf

Warning in min(col, na.rm = TRUE): no non-missing arguments to min; returning
Inf
```

```r
  # Print normalized dataframe
  df
```

```
           a         b         c  d
1  0.0000000 0.0000000 0.0000000 NA
2  0.1111111 0.1111111 0.1111111 NA
3  0.2222222 0.2222222 0.2222222 NA
4  0.3333333 0.3333333 0.3333333 NA
5  0.4444444 0.4444444 0.4444444 NA
6  0.5555556 0.5555556 0.5555556 NA
7  0.6666667 0.6666667 0.6666667 NA
8  0.7777778 0.7777778 0.7777778 NA
9  0.8888889 0.8888889 0.8888889 NA
10 1.0000000 1.0000000 1.0000000 NA
```

B. Next improve the below example code for the analysis of protein drug interactions by abstracting the main activities in your own new function. Then answer questions 1 to 6 below. It is recommended that you start a new Project in RStudio in a new directory and then install the bio3d package noted in the R code below (N.B. you can use the command `install.packages("bio3d")` or the RStudio interface to do this). Then run through the code to see if it works, fix any copy/paste errors before simplifying to a core working code snippet, reducing any calculation duplication, and finally transferring it into a more useful function for you.

```r
# Can you improve this analysis code?
library(bio3d)
s1 <- read.pdb("4AKE") # kinase with drug
```

```
Note: Accessing on-line PDB file
```

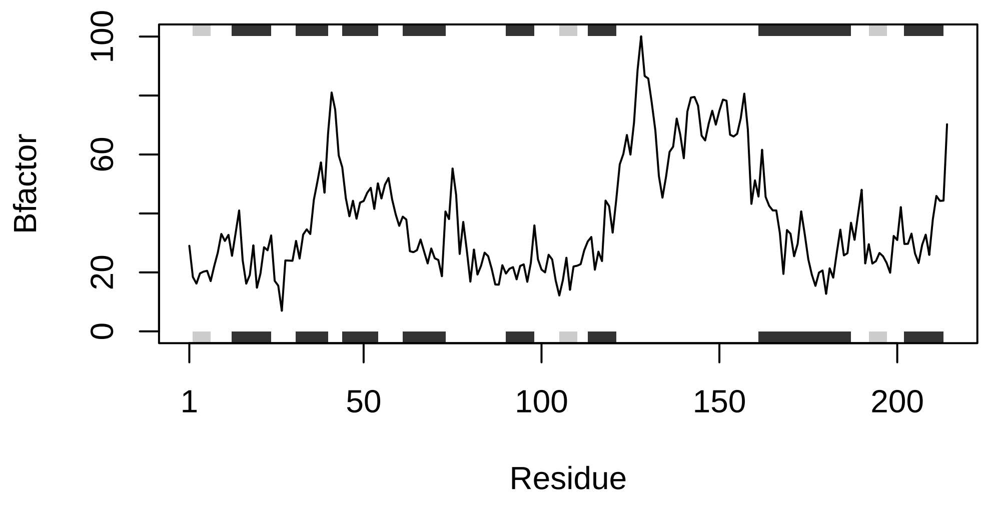
```r
s2 <- read.pdb("1AKE") # kinase no drug
```

```
Note: Accessing on-line PDB file
 PDB has ALT records, taking A only, rm.alt=TRUE
```

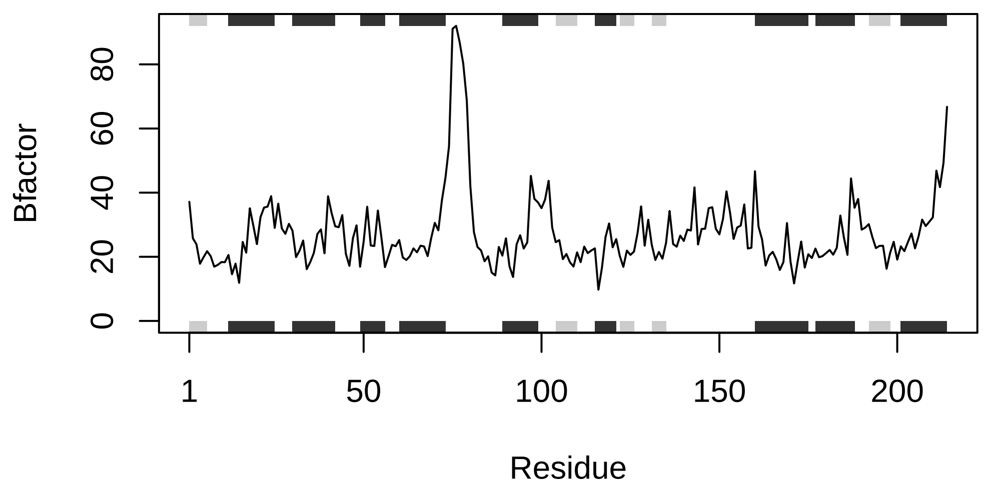
```r
s3 <- read.pdb("1E4Y") # kinase with drug
```

```
Note: Accessing on-line PDB file
```

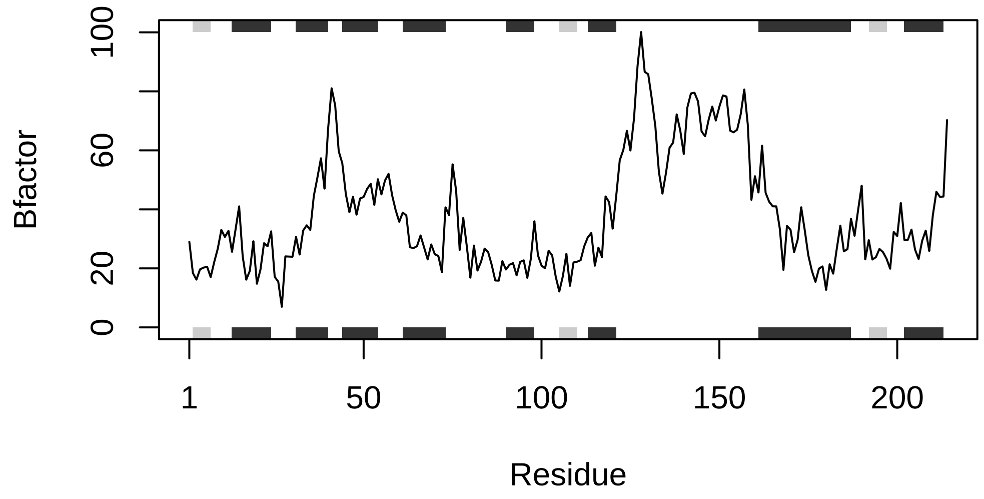
```r
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s1, chain="A", elety="CA")
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```



```
plotb3(s2.b, sse=s2.chainA, typ="l", ylab="Bfactor")
```



```r
plotb3(s3.b, sse=s3.chainA, typ="l", ylab="Bfactor")
```



Q1. What type of object is returned from the read.pdb() function?

read.pdb() function returns an object from the class "pdb" that has the Protein Data Bank data inside.

Q2. What does the trim.pdb() function do?

The `trim.pdb()` function works to trim an object from the PDB class so that it only shows the specified parts of the protein structure.

Q3. What input parameter would turn off the marginal black and grey rectangles in the plots and what do they represent in this case?

In order to turn off the marginal black and grey rectangles in the plot, I would set `mar = F` and in this case they represent structure elements in the protein.

Q4. What would be a better plot to compare across the different proteins?

A scatter plot would probabaly be better to compare different proteins.

Q5. Which proteins are more similar to each other in their B-factor trends. How could you quantify this?

Q6. How would you generalize the original code above to work with any set of input protein structures?

To generalize the original code from above to work with any set of protein structures, I would create a function that accepts a vector of PDB. I would then make it follow through the code reading the PDB files, trimming the PDB, extracting B-factors and then of course finally plotting them. This would work for any st of proteins.